1 Forelesning 14

1.1 Rettede grafer

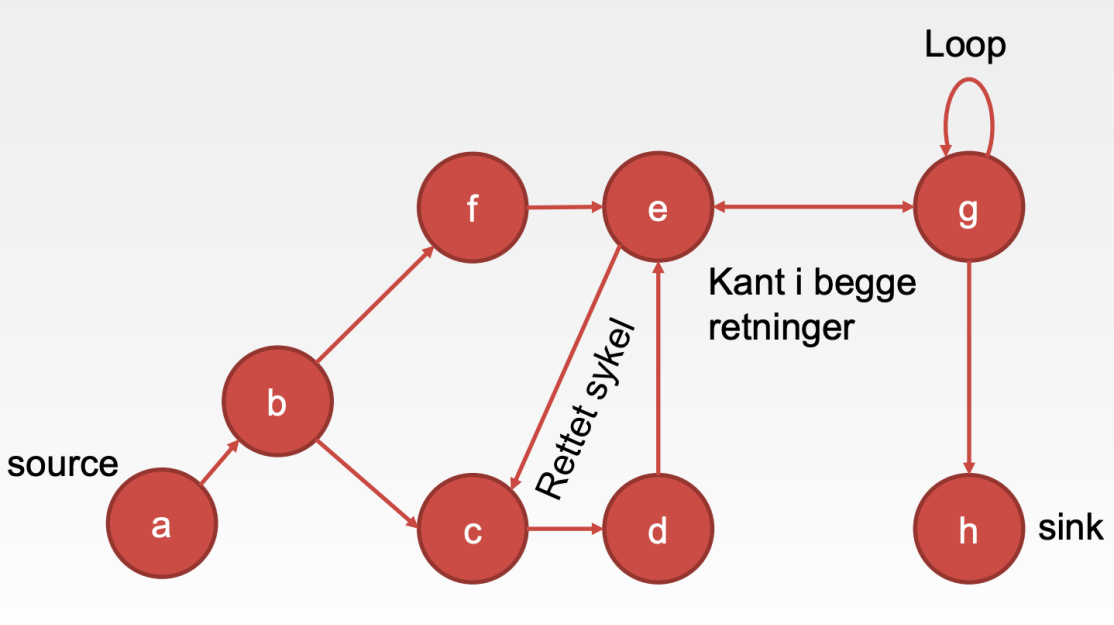

Figure 1: Eksempel på en rettet graf

Noder som kun har utgående kanter kalles `source`, og noder som kun har inngående noder kalles `sink`.

1.1.1 Nye metoder i rettede grafer

- `Iterable<V> outNeighbours (V v)`
- `Iterable<V> inNeighbours (V v)`
- `int outDegree`

1.1.2 BFS og DFS i rettet graf

- Virker akkurat som i urettet graf
- Der du før brukte `neighbours()`, må du nå bruke `outNeighbours()`
- DFS kan burkes til å finne en rettet sykel
  - Ser dere hvordan?
  - Jo, hvis vi kommer til en node som allerede ligger i `found`.

1.2 Strongly connected components

- Forkortes SCC
- Det finnes sti fra a til b og sti fra b til a, hvis og bare hvis a og b er i samme SCC.

- Det vil si at en strongly connected component er et sett med noder der det alltid finnes en sti du kan følge for å komme dit du vil.

1.3 Directed Acyclic Graph

- En graf uten noen rettede sykler kaller vi en DAG.
- Alle nodene i en DAG kan ordnes slik at alle kanter går fra venstre til høyre.
- Hvordan kan vi finne en slik ordning?

1.3.1 Topological sort (finn ordning)

```
int i = 0
while (g has a source)
    let v be a source
    order(i) = v
    i++
    remove v from g
```

1.3.2 Topological sort kjøretid

- Kan vi finne neste source fortere enn O(n)?
  - Ja, ha en liste over alle som har gradtall 0.
  - Hver gang gradtall oppdateres
    - Hvis ny verdi er 0, legg til i liste
  - Total $O(m)$
    - Oppdatering $O(m)$
    - $O(n)$ flyttinger til/fra liste